EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Exploiting programmable architectures for WiFi/ZigBee inter-technology cooperation

Peter De Valck[1*], Ingrid Moerman[1], Daniele Croce[2†], Fabrizio Giuliano[2†], Ilenia Tinnirello[2], Domenico Garlisi[2], Eli De Poorter[1] and Bart Jooris[1]

Abstract

The increasing complexity of wireless standards has shown that protocols cannot be designed once for all possible deployments, especially when unpredictable and mutating interference situations are present due to the coexistence of heterogeneous technologies. As such, flexibility and (re)programmability of wireless devices is crucial in the emerging scenarios of technology proliferation and unpredictable interference conditions.

In this paper, we focus on the possibility to improve coexistence performance of WiFi and ZigBee networks by exploiting novel programmable architectures of wireless devices able to support run-time modifications of medium access operations. Differently from software-defined radio (SDR) platforms, in which every function is programmed from scratch, our programmable architectures are based on a clear decoupling between elementary commands (hard-coded into the devices) and programmable protocol logic (injected into the devices) according to which the commands execution is scheduled.

Our contribution is two-fold: first, we designed and implemented a cross-technology time division multiple access (TDMA) scheme devised to provide a global synchronization signal and allocate alternating channel intervals to WiFi and ZigBee programmable nodes; second, we used the OMF control framework to define an interference detection and adaptation strategy that in principle could work in independent and autonomous networks. Experimental results prove the benefits of the envisioned solution.

Keywords: MAC protocols; Protocol prototyping; Wireless network control; Cognitive networks; Dynamic MAC adaptation; Cross-technology interference; Interference avoidance

1 Introduction

Recent years have witnessed an increasing adoption of heterogeneous technologies operating in unlicensed industrial, scientific, and medical (ISM) bands, thereby creating serious problems of coexistence and spectrum overcrowding. Although most wireless technologies (such as ZigBee, Bluetooth, and WiFi) have been designed to work in the presence of interference, it has been observed that performance may degrade significantly because of heterogeneous sensitivity to detect or react to the presence of other nodes and technologies.

While standardization groups are continuously defining standard amendments and extensions devised to cope with novel coexistence scenarios, the wireless research and academic community has pushed forward the vision of device programmability (started long time ago with cognitive [1] and active [2] networks) to cope with unpredictable and mutating interference situations, adapt to service demand variations and smartly exploit temporarily unused radio spectrum. In a fully programmable vision, the protocol stack defined in each standard should not be designed once for all possible deployments, but the *most appropriate* protocol fitting each *specific* context should be automatically employed when needed. In this vision, protocols would be simpler (for instance, why bother with hidden terminals in contexts where they are not present?), and backward or cross-technology compatibility would not be an issue any more.

In this paper, we specifically deal with ZigBee and WiFi technologies. Despite the fact that many mechanisms have been included in the relevant IEEE 802.15.4 and IEEE

*Correspondence: peter.devalck@intec.ugent.be
†Equal contributors
[1]Ghent University - IBCN - iMinds, Gaston Crommenlaan 8, 9000 Ghent, Belgium
Full list of author information is available at the end of the article

802.11 standards to cope with interference (e.g., carrier sense, adaptive modulation and coding, signal spreading), both technologies can seriously suffer in the presence of each other [3]. Even when sufficient resources (spectrum and time) are available, coexistence problems arise because of lack of coordination due to heterogeneous carrier sense granularity [3] and operating conditions not explicitly considered by the standards.

To remedy this situation, we propose to exploit the programmable architectures recently introduced for both technologies, namely the Wireless MAC Processor (WMP) [4] for WiFi and SnapMAC [5] for ZigBee, which are powerful and versatile tools for dynamically changing the MAC characteristics of the devices. We introduce a cross-technology coordination mechanisms based on the detection of cross-technology interference and on-the-fly adaptations of the MAC protocol rules, in order to accommodate both technologies on the same interfering channels in a TDMA-like fashion. We will adapt the standard MAC behavior of IEEE 802.15.4 and IEEE 802.11 devices but will keep referring to them as ZigBee and WiFi devices for the remainder of this paper.

2 Related work

Several analytic and simulation models, as well as experimental studies, have been proposed for characterizing the cross-technology interference in ZigBee and WiFi networks [3,6-8]. While most studies focus on the analysis of ZigBee performance degradation in the presence of WiFi interference, recently, it has been shown that significant throughput reductions can also be observed in WiFi networks [3,9]. Surprisingly, WiFi vulnerabilities arise despite the fact that many mechanisms have been included at the medium access control (MAC) and physical (PHY) layer to guarantee robustness to interference. The phenomenon has been clarified by considering two different main reasons. (i) An intrinsic reason: vendor-dependent implementation choices in some cases make it difficult to detect non-WiFi modulated signals or introduce latency in the receiving chain [10]. (ii) An extrinsic reason: due to the longer sensing time required by ZigBee to detect channel activity, it can not always detect WiFi packets to prevent collisions [11,12].

A first important requirement for proposing adaptation mechanisms in case of cross-technology interference is identifying the presence of two overlapping ZigBee and WiFi networks. The monitoring of heterogeneous radio frequency (RF) signals on ISM bands has been specifically addressed in [13]. This paper describes a monitoring infrastructure based on GNU Radio that is able to identify different technologies and demodulate received signals with the corresponding receiver modules. Although this approach is very effective, it requires additional dedicated hardware. The possibility to identify WiFi signals by using commodity ZigBee nodes has been explored in [14] and [15]. The approach proposed in [14] is based on the analysis of temporal samples of link quality indicators and received signal strength indication (RSSI) values, as well as on the identification of the portions of ZigBee corrupted packets by comparing those with typical WiFi transmission times. A similar temporal analysis is carried out in [15] with the aim to find periodic interference signatures caused by WiFi beacons and enabling the detection of WiFi networks by using a low-power monitoring interface. The possibility to detect ZigBee and other interference sources by means of WiFi commodity cards is explored in [16] by using an 802.11n PHY able to read RSSI values at different sub-carriers for characterizing spectral, energy, and pulse signals that are mapped into a technology classification scheme.

Once interference is detected, it is often required to make non-overlapping ZigBee and WiFi transmissions. However, solutions that simply choose a better channel upon the detection of interference are becoming nonviable because of the increasing number of technologies and applications in the market. Advanced solutions rely on complex and expensive radio transceivers to communicate with multiple protocols and different technologies [17], or increase the complexity of the transmission by using error correction codes or multiple antennas [18]. In addition, different approaches have considered the possibility to introduce some indirect forms of coordination between the two technologies. A time-based coordination is proposed in [10], where the authors first characterize the WiFi idle intervals in a given network scenario (called *white spaces*) and then adapt the ZigBee frame lengths and transmission intervals in order to maximize the probability to transmit during these white spaces. Channel reservations have been achieved in [11] by using two ZigBee channels overlapping with the WiFi one: the first channel is adapted to transmit a busy tone able to activate the WiFi carrier sense, while the second channel is used by the other ZigBee nodes to transmit their data in parallel to the busy tone transmission (i.e., while WiFi stations are prevented from accessing the channel). Other schemes introduce some simple forms of adaptive redundancy (e.g., by repeating the header in the ZigBee packet [12], whose transmission time is comparable with a WiFi packet transmission time) for improving the ZigBee resilience to WiFi transmissions that typically overlap at the beginning of the ZigBee frame because of the limits in the ZigBee carrier sense granularity.

These coordination solutions require a customized tuning of the MAC parameters used in the multi-technology interfering networks or even modifications of the medium access operations. However, making these modifications context-dependent and dynamic is not viable on *off-the-shelf nodes*: on one side, in WiFi nodes, low-level MAC

operations are hard-coded into the network cards for efficiency reasons and only support a parametric control model (e.g., tuning of the contention windows); on the other side, even when the MAC is implemented in software as in the case of ZigBee nodes, dynamic modifications cannot be supported at run-time, without recompiling and reloading the relevant software modules.

3 Programmable MAC architectures

As stated before, support for flexibility and (re)programmability of wireless devices is crucial to cope with emerging scenarios of technology proliferation and unpredictable interference conditions. Wireless communication performance should be matched to the demands of the application being used, and should be able to intelligently exploit opportunistically available spectrum and resources in dense environments. While pursuing such an idealized, adaptive vision, the research community has mainly worked on programmable radio platforms, such as GNUradio [19], WARP [20], Airblue [21], etc. However, very recently, a different paradigm has been proposed for trading-off flexibility and ease of programmability by identifying suitable abstractions and relevant programming languages devised to formally describe wireless protocols. Such a paradigm has lead to the definition of novel programmable architectures for technologies such as WiFi and ZigBee, in which elementary commands are composed in a protocol logic defined in terms of state machines or command chains, on the basis of the following decoupling compromise:

- wireless cards support a hard-coded (not modifiable by the MAC protocol programmer) *instruction set*, namely an application programming interface (API) comprising of elementary commands and signals;
- third-party MAC programmers describe how commands are coordinated and triggered in a formal (executable) model, such as a state machine or a command chain.

These architectures enable dynamic network optimization that can be executed by a network controller by loading the most suitable protocol on the fly on the basis of a context estimate.

3.1 Wireless MAC processor

While each protocol (or protocol release) supports different features and each vendor implements a specific hardware/software card architecture, there is an interesting common set of *capabilities* and *functions* that can be exploited for defining hardware agnostic programs to be loaded on the wireless cards and change their behavior. The Wireless MAC Processor [4], whose architecture is shown in Figure 1, is based on this observation. The core of the architecture is an execution engine capable of running programs defined as extended finite state machines (XFSMs). These state machines are composed of the following: a set of signals provided by the hardware subsystems by means of an interruption block, a set of elementary functions implemented into an operation block, and a set of registers to save system state and configuration parameters. A memory block is dedicated

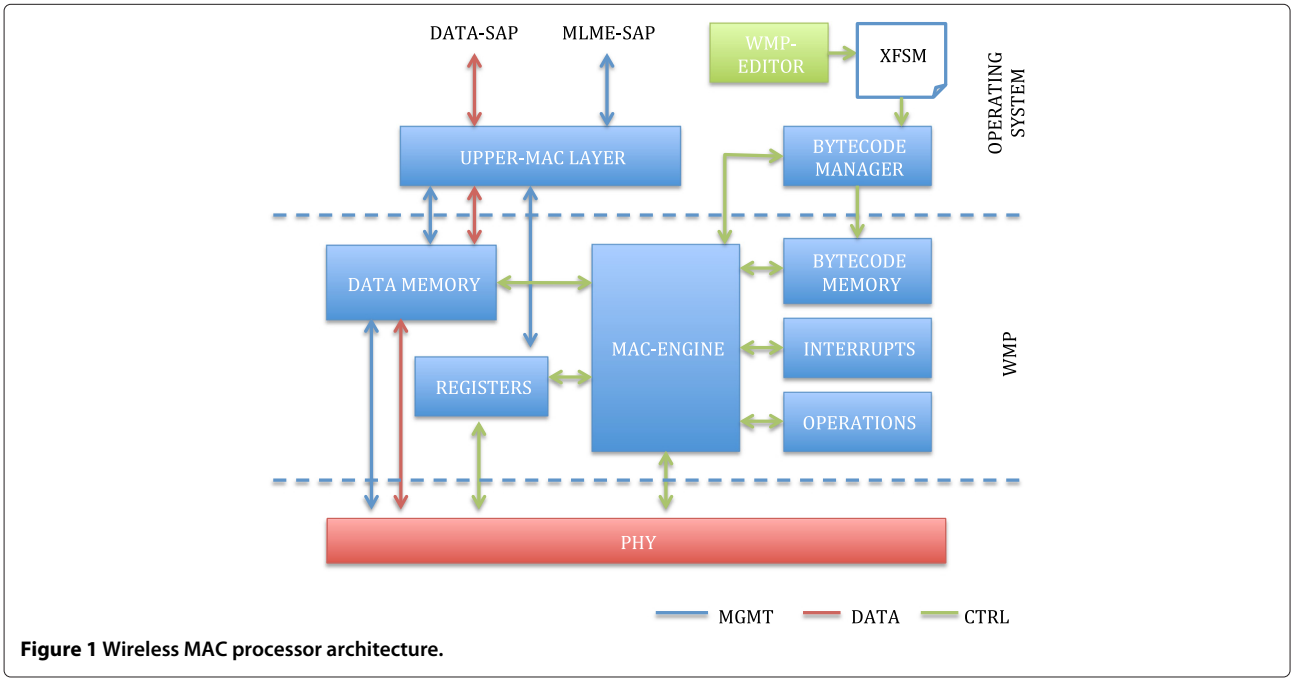

Figure 1 Wireless MAC processor architecture.

to the storage of MAC programs (XFSMs), while a control interface is available for loading programs and tuning configuration parameters.

According to this architecture, the card does not implement a standard-specific predefined protocol, but it acts as a generic executor of state machines reacting to internal events of the system (e.g., the arrival of a new packet from the host) or external events of the channel (e.g., the reception of a new packet from the air interface). The reactions to the same signals may vary according to the system state, which includes the *state of the hardware* and the *logical state* of the programmed protocol.

3.1.1 The WMP application programming interface

A breakdown analysis of MAC protocols reveals that they are well-described in terms of three types of elementary building blocks: *actions*, *events,* and *conditions*.

Actions are commands acting on card hardware. In addition to ordinary arithmetic and memory related (*set/get*) operations, which work on system registers and queues, dedicated actions implement atomic MAC functions such as transmit a frame, set a timer, write a header field, switch to a different frequency channel, etc. Actions are not meant to be programmable. As the instruction set of an ordinary central processing unit (CPU), they are provided by the hardware vendor. The set of actions may be extended at will by the device vendor and can also include advanced operations on the PHY, such as the configuration of the physical channel and the selection of the desired encoding scheme.

Events include hardware interrupts such as channel up signals, indication of reception of a valid preamble or end of a valid frame, expiration of timers, and signals conveyed from the higher layers such as the enqueuing of a new packet. As in the case of actions, the list of supported events is a-priori provided by the hardware design.

Conditions are boolean expressions evaluated on internal configuration and statistic registers. These registers are either explicitly updated by actions or implicitly updated by events. Some registers are dedicated to store general hardware parameters whose tuning automatically affects the hardware configuration (such as the operating channel, power level, transmission format, selected packet queue), while some others store programmable data related to enqueued frames (source or destination address, frame size, etc) or MAC protocol variables (contention window, back-off counter, etc.) Registers are used to provide an interface to the PHY layer and to achieve a more compact protocol description.

Actions, events, and registers on which conditions may be set, form the application programming interface exposed to third party programmers. This API is implemented (in principle) once-for-all, meaning that programs may use such building blocks to compose a desired operation, but have no means to modify them.

Figure 2 shows a simple example of state machine defined on the basis of elementary events, actions, and conditions (using self-explanatory names). The program can be coded into a table listing all possible state transition relations, an initial state, and an initial set of hardware configuration parameters.

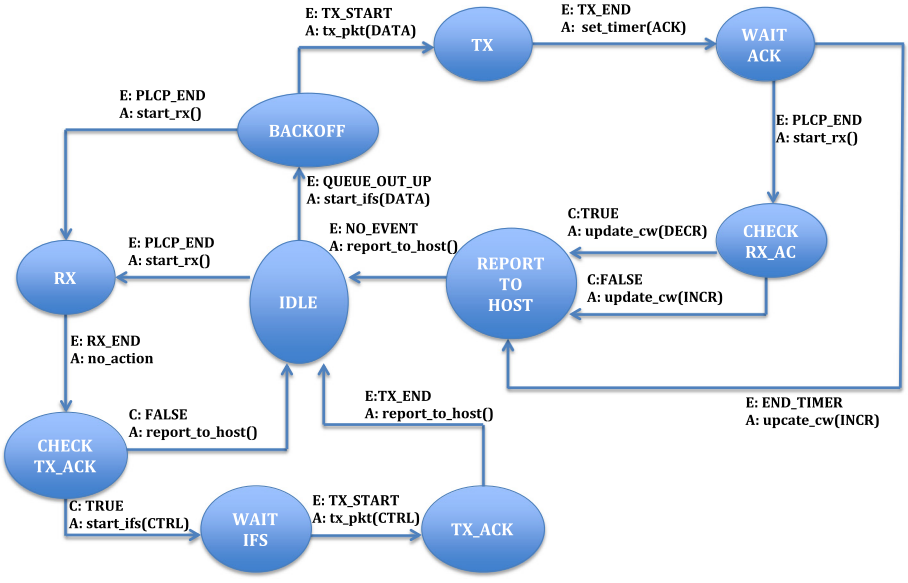

Figure 2 A simple XFSM defining legacy DCF operations.

3.1.2 MAC engine: the CPU

The ability to timely react to events is a crucial property of lower-MAC protocols (e.g., for triggering a transmission right at the end of a timer expiration). In the Wireless MAC Processor architecture, this is accomplished by implementing an XFSM execution engine, called MAC engine, directly on the radio hardware. The MAC program, namely the tables containing all the possible state transitions, is loaded in a memory space deployed on the hardware. Starting from the initial state and parameters of the selected memory slot, the MAC engine fetches the table entry corresponding to the state, reads the list of all the possible events triggering a transition from that state, and loops until one of these events occurs. It then evaluates the associated conditions on the system parameters, and if this is the case, it triggers the associated action and parameters' updates (if any), executes the state transition, and fetches the new table entry for such destination state.

3.2 SnapMAC

One of the major issues with the radio drivers for sensor nodes is the lack of strict timing control (since the same CPU controls the radio and the user code) and the lack of flexibility. The first is a major problem when developing time sensitive protocols (such as TDMA, fast software acknowledgements,...) while the second prohibits swapping or modifying protocols at run-time. The SnapMAC architecture [5] was designed to allow easy implementation and modification of novel MAC protocols on sensor nodes without compromising the performance.

In contrast to traditional systems, where protocol logic is hard-coded, SnapMAC uses the concept of dynamic "*chains*" to define protocol logic. These chains are composed of small commands that each execute a single task. All information on these commands resides in the command pool, where commands are further grouped into modules, depending on the type of functionality they provide. Radio commands operate directly on the radio (turn on the radio, transmit a packet), memory commands perform memory operations (move buffer, clear memory), arithmetic commands can be used to manipulate variables (increment, decrement), etc. Finally, the flow commands can be used to control execution flow through the chain. By combining these commands, it is possible to create complex chains that execute the logic of any MAC protocol.

An overview of the SnapMAC architecture can be seen in Figure 3. Two interfaces are offered to the user code: a data interface and a control interface. The data interface is used to exchange buffers that can contain data to transmit, received packets, or even spectrum-sensing information. The control interface is used to compose or modify chains and control their execution.

When activating a chain, it is possible to specify a time at which a specific command has to be executed. The scheduler will then calculate the execution time of all earlier commands to ensure that the requested timing is met. Thanks to this functionality, it is possible for MAC designers to define time critical protocols without the need to know hardware specific timing information. The scheduler will instruct the command dispatcher when to execute each command.

A simple SnapMAC chain is shown in Figure 4. This chain will transmit a single packet on a specific channel

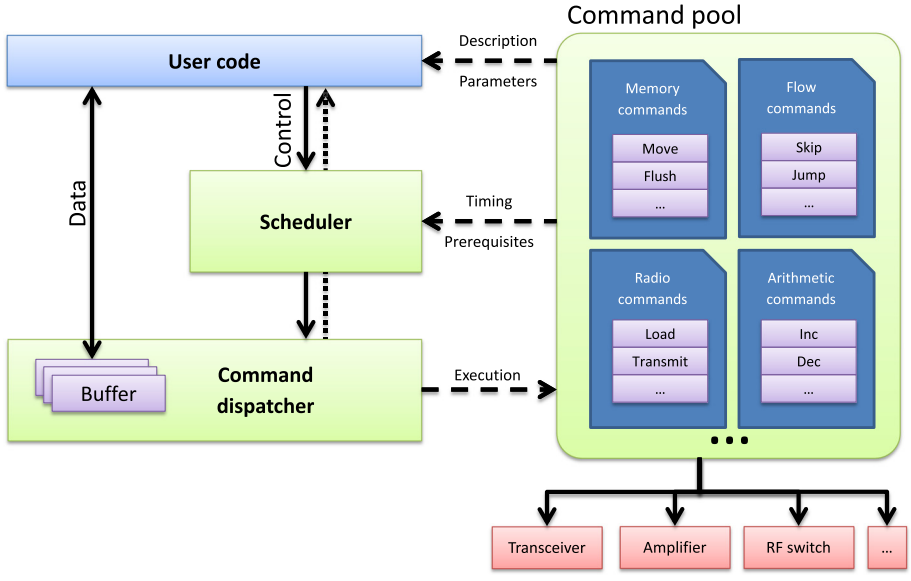

Figure 3 Overview of the SnapMAC architecture.

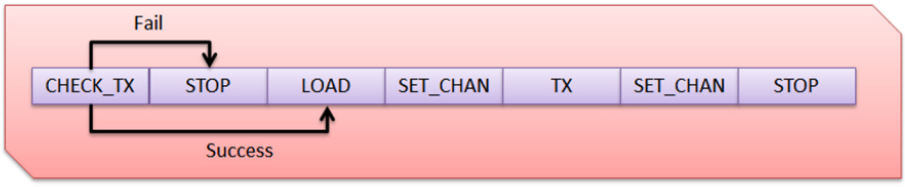

Figure 4 A simple SnapMAC chain. This simple chain can be used in the SnapMAC architecture to transmit a single frame.

and then return to the original channel. Upon activating this chain, the time at which the packets must be transmitted can be specified. When the chain is finished, the stop command can periodically reschedule this chain at a later time. This way, a simple TDMA scheme can be implemented without having to know the time it takes to change the channel or load the packet into the radio.

4 WiFi/ZigBee coexistence: problems and coordination strategy

4.1 Cross-technology interference

Although WiFi and ZigBee medium access protocols utilize CSMA to avoid packet collisions in the ISM unlicensed bands, they do not explicitly take into account the peculiarities of potential interfering technologies. Despite the use of different channel bandwidths (20 MHz for WiFi, 2 MHz for ZigBee) and transmission powers (up to 20 dBm for WiFi, up to 5 dBm for ZigBee), both technologies generally suffer from mutual interference when they are co-located in the same environment.

One of the major reasons of this performance degradation is the different granularity at which clear channel assessment (CCA) samples are collected. Specifically, a ZigBee node spends 128 μs sensing channel activity and 192 μs switching from reception to transmission mode. If a WiFi node (whose back-off slot is only 9 μs) starts a transmission during this switching time, it will not be detected by the ZigBee node resulting in packet collisions (as shown in Figure 5 where the measured channel power is measured with μs resolution). In addition, as ZigBee frames are transmitted with a low data rate, they occupy the medium for a long time and thereby defer WiFi transmissions. As a result, if a ZigBee node is transmitting at sufficiently high-power levels or if a ZigBee node is located close to a WiFi node, it may degrade the WiFi performance drastically.

Table 1 shows the effect of cross-technology interference on a WiFi and ZigBee link with two constant transmitters. The ZigBee transmitter is located about 2 m from the WiFi receiver. The percentages shown in the table are calculated by dividing the transmission data rate by the measured throughput. Even without interference, several factors cause this throughput to be significantly lower than the advertised data rate. For WiFi, the low data rate at which the header is transmitted is one of the major bottlenecks [22], while CCA and RX-TX switching cause the lower throughput of ZigBee transmissions. From the table, it is evident that even using robust modulation schemes (namely, 1 Mbps), WiFi throughput can be reduced to less than 50% of the interference-free

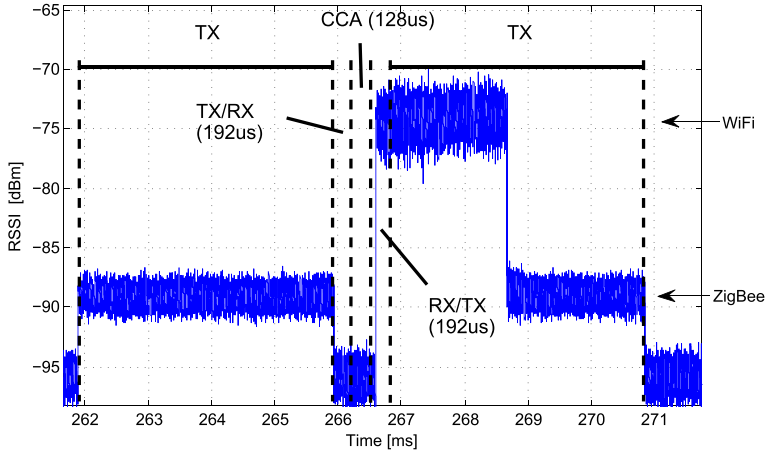

Figure 5 Collision between WiFi and ZigBee. A WiFi frame interferes with a ZigBee frame because the WiFi radio senses the channel as idle during the long RX-TX turn around time of the ZigBee radio. After this turn around, the ZigBee radio will also start transmitting; but due to the higher transmit power of WiFi, this cannot be seen in the figure.

Table 1 Actual obtained throughput and % of the theoretical throughput of WiFi and ZigBee with and without cross-technology interference

	Scenario	WiFi at 18 Mb/s		WiFi at 6 Mb/s		WiFi at 1 Mb/s	
WiFi	WiFi only	9.91 Mb/s	(55%)	4.45 Mb/s	(74%)	887.88 Kb/s	(89%)
	WiFi + ZigBee	117.6 Kb/s	(0.7%)	2.16 Mb/s	(36%)	442.96 Kb/s	(44%)
ZigBee	ZigBee only	194.79 Kb/s	(78%)	194.79 Kb/s	(78%)	194.79 Kb/s	(78%)
	ZigBee + WiFi	52.66 Kb/s	(21%)	47.93 Kb/s	(19%)	39.91 Kb/s	(16%)

throughput. Using a higher data-rate, the effect is even more significant and WiFi throughput is almost non-existent. The influence of WiFi interference on the ZigBee throughput is also significant with a consistent reduction to about 1/4 of the interference-free throughput.

4.2 Cross-technology TDMA

The easiest way to avoid interference between two technologies is to operate both technologies on non-overlapping channels. However, given the increased number of wireless devices, finding free channels is not always feasible. In such a situation, it can be advantageous that the two technologies share the same channel with some form of coordination to avoid collisions.

Several coordination strategies have been proposed (as briefly reviewed in Section 2), but none of these solutions are capable of adapting the MAC strategy dynamically to the context. The WMP and SnapMAC programmable architectures can easily support run-time adaptations to the medium access operations. In particular, we propose a cross-technology TDMA scheme in order to separate the transmission intervals of WiFi and ZigBee and avoid cross-technology interference. Channel time is split into periodic frames in which we alternate ZigBee and WiFi transmissions. ZigBee nodes are entitled to transmit for a portion of the frame equal to t_{ZigBee}, while WiFi nodes can access the channel (according to the legacy distributed coordination function (DCF)) in the following t_{WiFi} interval. Figure 6 shows a picture of the cross-technology TDMA operations: ZigBee nodes autonomously switch between active and idle intervals (whose duration is set to t_{ZigBee} and t_{WiFi}) while being synchronized to the ZigBee coordinator. WiFi nodes switch to the activity interval after the detection of a burst of consecutive ZigBee packets and go to idle at the expiration of a t_{WiFi} timer.

This cross-technology TDMA scheme can be implemented by the following: i) forcing all nodes to cease transmitting during the time interval allocated to the other technology and ii) introducing a synchronization signal that can act as a common temporal reference for all nodes. The first feature can be added to the legacy DCF state machine (using the WMP) by scheduling a timer expiration event that triggers the transition toward a waiting state and to the ZigBee command chain (for SnapMAC) by stopping and re-scheduling it for execution at a later time. To synchronize transmission intervals between the different technologies, the adopted solution uses the ZigBee transmissions as a temporal reference for both technologies. The ZigBee transmission chain will automatically be re-activated after the WiFi transmission interval. WiFi nodes will then use the ZigBee transmissions as a reference signal to determine the start of their transmission interval. This is a feasible approach because when ZigBee interference is harmful for WiFi transmissions (i.e., the interfering power is high enough to be detected by WiFi), it has been shown [23] that WiFi cards are triggered by ZigBee transmissions, causing different types of errors which can be analyzed and

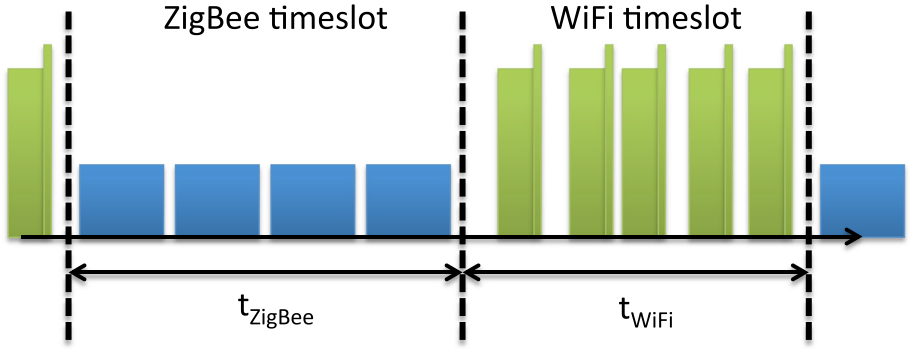

Figure 6 TDMA scheme for WiFi and ZigBee coexistence. Instead of randomly accessing the channel, each transmitter will limit its transmissions to the assigned time slot.

compared with the error patterns typical of WiFi transmissions. In case of ZigBee interference signals, errors may appear randomly at any point during the time the WiFi demodulator is active, while for WiFi modulated signals error statistics vary during frame reception and depend on frame length and modulation used. By measuring the occurrence rate of physical layer convergence procedure (PLCP) errors, checksum failures, and invalid headers and the duration of the error bursts it is possible to detect the presence and the duration of non-WiFi transmissions. For example, Figure 7 shows the errors generated by four ZigBee packets on the WiFi network card. In the particular implementation of the transceiver used in the experiments, the reception of the ZigBee frame generates one error event approximately every 1 ms (plus one for the acknowledgment). Combining the presence of several bad PLCP errors (following the statistics of non-WiFi modulated signals) together with the channel busy time information provided by the card, we are able to detect: i) the beginning of a burst of ZigBee packets and ii) the end of this burst and the beginning of the WiFi transmit slot.

5 Using OMF as an inter-technology control framework

For the evaluation of the above solutions, the Orbit Management Framework (OMF) and Measurement Library (OML) were used. This framework has been designed for the configuration, execution, and centralized control of network experiments and is now deployed in several testbeds worldwide (from the USA to Australia), including the European CREW testbeds [24]. It is based on a client-server architecture: resource control (RC) processes running on the testbed nodes interact with a central experiment controller (EC) and database. Based on an experiment description file this EC instructs the RC processes to launch applications, generate traffic flows, collect performance information, monitor the node's status, re-configure the nodes when network events occur, etc.

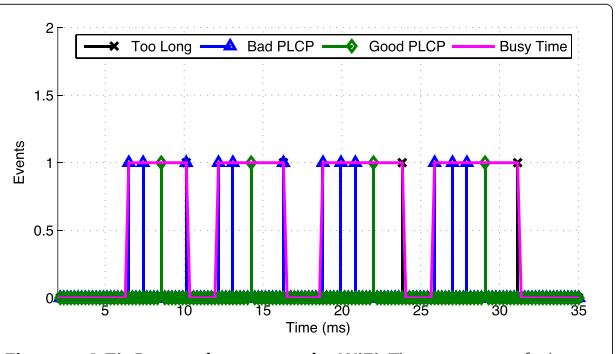

Figure 7 A ZigBee packet as seen by WiFi. The reception of a burst of errors and the analysis of the busy time is used by WiFi to detect the presence of ZigBee packets.

Although the framework has been designed for experiment control, following the approach proposed in [25], we exploited the OMF primitives to define a cross-technology MAC adaptation strategy that, in principle, could work in independent and autonomously evolving wireless networks. To this purpose, we integrated some WiFi and ZigBee nodes in the CREW testbed based on the WMP and SnapMAC architecture. Moreover, we developed an OMF wrapper for both the WMP and SnapMAC control interfaces to allow the EC to invoke the control commands (e.g. for dynamically loading or activating a new MAC state machine or command chain).

5.1 Supporting a MAC cognitive cycle

We consider a MAC cognitive cycle in which: i) the sensing phase is implemented by collecting throughput and error statistics by means of dedicated monitoring applications deployed on the nodes; ii) the analysis and reasoning phases are performed at the EC by aggregating data and defining customized events to be fired when cross-technology coexistence problems arise, iii) the adaptation phase is finally achieved by loading and/or activating cross-technology TDMA programs on the controlled nodes when needed. The cycle is depicted in Figure 8. ZigBee and WiFi receivers report the achieved throughput to a central database using the OML framework; WiFi nodes also report their error occurrence statistics. This data is continuously queried by the experiment controller (EC): when throughput reduction is observed for both technologies or when the WiFi error statistics correspond with an interfering ZigBee network, the EC triggers the run-time MAC reprogramming and the cross-technology TDMA programs (defined in terms of an XFSM for WiFi nodes and a command chain for ZigBee nodes) are activated on the controlled nodes.

Note that, for our WMP and SnapMAC architectures, reprogramming a MAC protocol does not require the loading of a new software module and/or rebooting the nodes, because it is achieved by changing a parameter in the transition table (for the WMP) or command chain (for SnapMAC). Therefore, MAC reprogramming is transparent to applications and can be performed at run-time with negligible latency (corresponding to the time required to load the new parameters).

5.2 Alternative control mechanisms

The use of OMF allowed us to rapidly prototype and evaluate the performance of our cross-technology TDMA scheme, including the use of the existing control network to report node statistics to the central controller. This implementation simulates ZigBee and WiFi gateways connected to a wired backbone, but even if no existing control

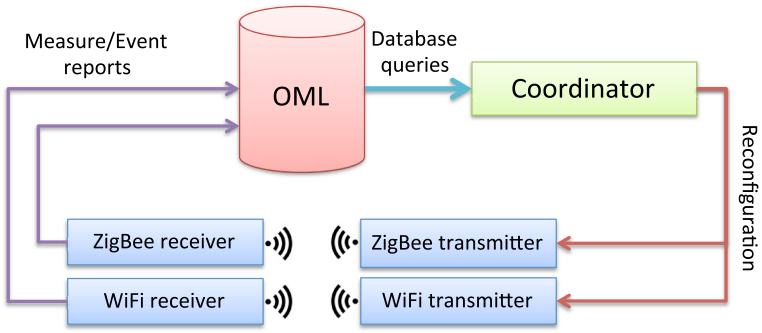

Figure 8 Cognitive cycle. The cognitive cycle as used for the experiments in this paper. Configuration messages are delivered using the OMF framework.

channel is available, several solutions are possible and are discussed in the following section.

5.2.1 Multi-technology nodes

Nodes equipped with both WiFi and ZigBee radios can act as centralized controllers or relay control traffic between the two different technologies as shown in Figure 9. This way, no additional hardware or physical channel is required. The control packets will have an impact on the application throughput, especially for low data-rate ZigBee communication, but this will be compensated by more efficient channel usage.

5.2.2 Inter-technology communication system

While both technologies are unable to decode each others packets, both are able to detect the presence of another technology. This fact is used by our TDMA implementation to synchronize the transmission intervals, but could also be used for more complex communication. For example, this technique is exploited in [26] to implement a novel inter-technology communication system, called BusyBee, that defines a modulation scheme and a message-oriented protocol that allows to exchange low-rate control information between WiFi and ZigBee nodes.

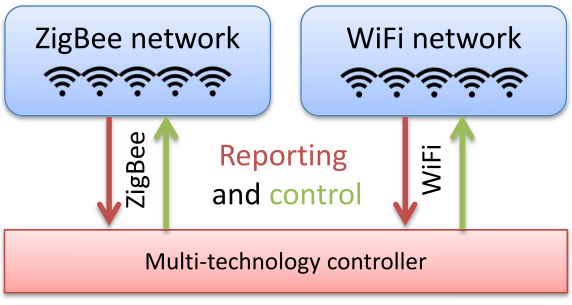

Figure 9 Control architecture using a multi-technology node. A single controller is able to control both technologies by using multiple interfaces.

A possible architecture using this method is shown in Figure 10.

5.2.3 Fully decentralized

In a fully decentralized architecture, each node can independently adapt its own operation by observing interference patterns, and no information is shared with a central controller, negating the need for a dedicated control channel. As the decision making is based on a more limited information set, this requires more intelligent adaptation strategies and can lead to sub-optimal results. As with the previous solutions, there is a trade-off between the reduced number of control messages and the throughput gain of the adaptations. While more complex, the use of flexible MAC architectures allowing runtime reconfiguration will simplify the implementation of these solutions.

Although these alternative solutions are feasible in terms of conceptual communication, OMF provides a

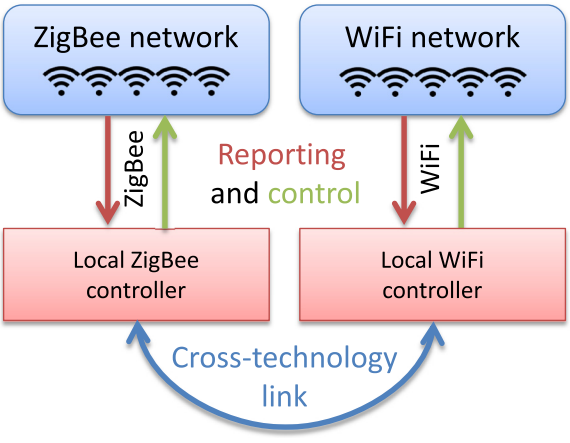

Figure 10 Control architecture using cross-technology communication. Both technologies are controlled by individual controllers linked by a cross-technology link.

simple way to collect measurements and detect interference patterns and was therefore used to implement our cognitive cycle.

6 Experimental results

We have tested the cross-technology TDMA scheme in the CREW w-ilab.t testbed [27], composed of over 60 nodes spread in a semi-shielded environment of 22 × 66 m^2. We selected one pair of WiFi nodes, Alix 3C3 embedded PCs equipped with a bcm4318 wireless interface running our custom-made WMP firmware [4], and one pair of interfering ZigBee nodes, RM090 [28] motes running TinyOS and using the SnapMAC driver. During the experiments, the WiFi nodes executed the legacy state machine and reported throughput and error statistics to the central database. When triggered by the experiment controller, they switched to the TDMA state machine. The ZigBee nodes were programmed with two SnapMAC chains: on the receiving node, a simple chain was responsible for receiving and reporting all ZigBee packets. On the transmitting node, a TDMA chain was running that could switch between regular CSMA operation and TDMA operation as instructed by the experiment controller.

The WiFi nodes used WiFi channel 11 (centered at 2462 MHz) for communication while the ZigBee nodes used ZigBee channel 23 (centered at 2465 MHz). Together with a transmit power of 15 dBm for WiFi and -5 dBm for ZigBee, this realistic setup guarantees that both technologies will interfere when they transmit at the same time. Without interference, the WiFi link is capable of reaching speeds of up to 4.5 Mb/s (using a data-rate of 6 Mb/s), and the ZigBee link can reach 195 Kb/s (using the fixed data-rate of 250 Kb/s). It should be noted that the speeds reported in the following experiments are averaged over a period of 1 s. As we use a TDMA period of 100 ms, the TDMA cycles themselves are not visible in these figures.

6.1 Functional analysis

In order to verify the effectiveness of the envisioned cross-technology coordination strategy, we used a USRP node to record RSSI samples with a very high temporal resolution (at a μs scale). This RSSI trace visualizes the channel access operations performed by both ZigBee and WiFi transmitter.

Figure 11 shows an example of such a trace, using legacy access protocols (top) and cross-technology TDMA coordination (bottom). The figure clearly shows the transmission intervals where the RSSI values are higher than the background noise. Because of the different transmission powers, the RSSI values corresponding to the WiFi transmissions are about 20 dB higher than ZigBee transmissions: the power level sensed for WiFi frames and WiFi acknowledgments are, respectively, -70 and -62 dBm, while the power level of ZigBee frames is about -87 dBm.

In the case of legacy access, some ZigBee and WiFi transmissions collide, resulting in a missing acknowledgment and the overlapping of transmission intervals (e.g., the trace pattern shown at 40 ms in Figure 11). In the case of cross-technology TDMA, the figure clearly shows that the two technologies operate orthogonally by alternating channel access periods. This figure also shows an idle time at the end of each period to make sure the last transmission of one period will not interfere with the first transmission in the next period. For a given period duration, this idle time can be up to about 4 ms at the end of the ZigBee period or about 1 ms at the end of the WiFi period and is equal to the time to transmit a single frame at their respective data rates. The length of this idle time could

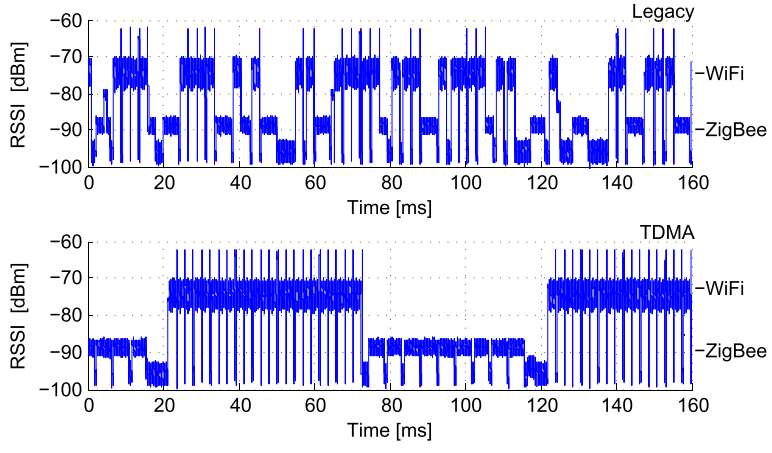

Figure 11 Channel access operations. Coexistence between ZigBee and WiFi nodes on the same channel in case of legacy channel access schemes (top figure) and cross-technology TDMA (bottom figure).

be reduced by transmitting variable length ZigBee packets, but all our experiments were performed using fixed length 120 byte ZigBee packets. Even with this modification a small guard interval would still be required to deal with synchronization offsets.

6.2 Performance analysis

The complete cognitive MAC cycle was tested by monitoring the throughput of the WiFi and ZigBee links while running the cognitive loop as described in Section 5.1. Instead of plotting the absolute throughput values, they are divided by their respective data rates to obtain normalized throughput values that can be displayed on the same scale.

The first experiment, shown in Figure 12, consists of a ZigBee link transmitting at a maximal throughput at the start of the experiment. This only corresponds to a normalized throughput of approximately 80%: the transmission of a ZigBee frame takes 4.2 ms, after which the radio requires at least 550 μs before it is able to start the transmission of the next frame. After 58 s, the WiFi link is activated with a data rate of 6 Mb/s, and the ZigBee throughput drops from 80% to about 20%. The decreased ZigBee performance in the presence of increased WiFi throughput is identified as cross-technology interference. After a few seconds, the TDMA scheme is activated, and both technologies are assigned equally sized slots. The ZigBee throughput returns to 35% while the WiFi interferer reaches a normalized throughput of 45%. This difference is caused by the longer idle time at the end of the ZigBee slot as shown in Figure 11. While this is less than the theoretical maximum of a 50% to 50% split, this is a significant improvement over the legacy situation where interference would also cause wasted channel time. By tuning the TDMA parameters, these throughput results can be equalized or tuned to specific traffic requirements as is demonstrated in the last experiment.

A second experiment was devised to monitor the packet error rate (PER) of the ZigBee link. For the entire time of the experiment, the ZigBee node transmits packets at a constant rate of 50 packets per second using the TDMA protocol. During the first part of the experiment, no interference is generated. During the second part, a WiFi interferer using legacy channel access is activated, and finally, this interferer is switched to the TDMA protocol using the same parameters as the ZigBee transmission. The results of this experiment can be seen in Figure 13, without interference the PER is close to 0%, but when the legacy interferer is activated, almost 30% of the packets are lost. After activating the TDMA scheme in the final part of the experiment, the PER for the ZigBee link again drops to 0%, and the WiFi throughput increases slightly. This indicates that a legacy interferer will cause almost 15% wasted channel time due to interference (about 30% of the ZigBee time slot). By using the TDMA protocol, this lost time is recovered, and the channel is used much more efficiently without negatively effecting the WiFi throughput.

A final experiment demonstrates the capability of the cognitive solution to change and distribute the TDMA parameters at run-time. Shown in Figure 14, the experiment starts with a ZigBee and WiFi transmitter without cooperation. After 20 s, the TDMA protocol is activated and at 35 and 50 s, the allocated TDMA slots are reconfigured dynamically. During the non-optimized operation, both technologies experience collisions resulting in throughput degradation and fluctuations. During the TDMA operation, throughput results are stable and depend on the allocated slot interval. For example, ZigBee slots are equal to 30% (28 ms for ZigBee, 64 ms for WiFi), 50% (50 ms for ZigBee and 44 ms for WiFi), and

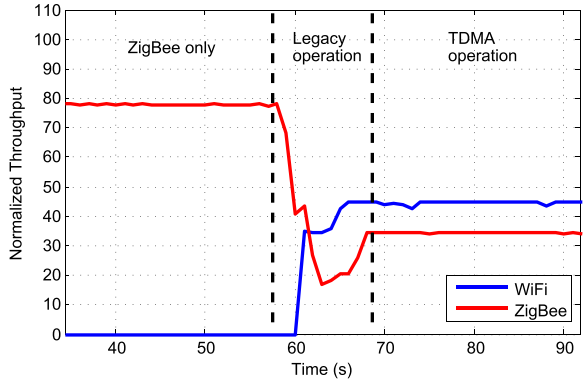

Figure 12 Comparison between legacy saturation traffic and TDMA. When both WiFi and ZigBee contend for the medium, overall throughput is dramatically reduced. Activation of our TDMA protocol improves both WiFi and ZigBee performance significantly.

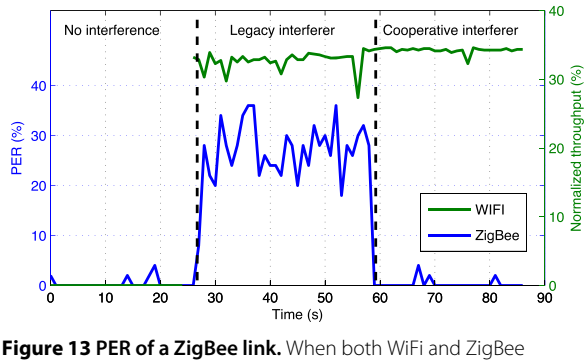

Figure 13 PER of a ZigBee link. When both WiFi and ZigBee contend for the medium, ZigBee PER increases dramatically. Using the TDMA protocol results in ZigBee PERs similar to the non-interfered situation without decreasing the WiFi throughput, indicating a more efficient channel usage.

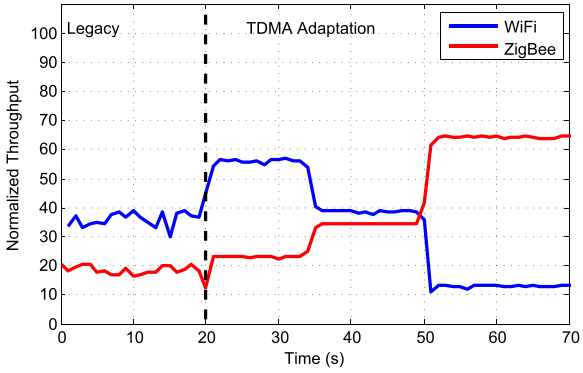

Figure 14 Adaptability of the TDMA parameters. The TDMA parameters of the WMP state machine and SnapMAC chain can be changed dynamically to accommodate for different WiFi and ZigBee traffic rates. The examples shown include a normalized throughput of about 23%, 35%, and 63% for the Zigbee traffic.

80% (79 ms for ZigBee and 14 ms for WiFi) of the TDMA period, respectively, resulting in a normalized throughput of about 23%, 35%, and 63%.

7 Future work

We currently do not implement the inverse adaptation cycle (from the execution of cross-technology TDMA scheme to the execution of legacy MAC operations). However, all components to support this kind of adaptation are already present and would only require extending the context detection of the coordinator and adding additional actions. Additionally, future work will also focus on further exploiting the capabilities of the re-configurable WiFi and ZigBee MAC architectures by designing cognitive solutions that can support multiple MAC protocols at the same time.

In order to test the feasibility of our cross-technology TDMA, we used a static distribution of the TDMA intervals, activated based on simple throughput measurements. As our flexible radio architectures are capable of reporting a wide range of cross-layer metrics, further research should evaluate the feasibility and impact of more advanced control algorithms that adapt the TDMA operation and interval based on this extended information base.

For the solution presented in this paper, we used the available OMF/OML infrastructure to enable the fast prototyping and verification of our cognitive solution. In a real deployment, this wired backbone would not be available but could easily be replaced by a wireless control channel to exchange configuration parameters and measurement information. Solutions such as multi-technology nodes acting as a controller, cross-technology signaling, or even completely decentralized operation can be easily implemented in the future by using flexible MAC architectures.

8 Conclusion

In this work, we have presented a dynamic coordination mechanism to improve the coexistence of ZigBee and WiFi networks in the emerging scenario of overcrowded ISM bands. In dense wireless environments, it is impossible for WiFi and ZigBee to operate in non overlapping channels and our experiments have shown that in such a situation both technologies undergo a severe throughput degradation.

The solution, explored in this paper, does not require dedicated hardware or modifications to the transmitter hardware for cross-technology interference detection, but only uses off-the-shelf ZigBee and WiFi hardware. The solution is based on the adoption of programmable MAC architectures to support run-time MAC protocol adaptations managed by a global network controller. In particular, we have proposed and experimentally verified a cross-technology TDMA protocol that can be used to efficiently share the spectrum between ZigBee and WiFi transmitters in overlapping frequency channels.

We have shown that experiment control infrastructure cannot only be used for experiment descriptions and experiment control, but also for cognitive network control. Using standard tools for experiment control, being the OMF for node management and the OML for data collection, a fully cognitive solution was created that is able to trigger protocol modifications upon the detection of cross-technology interference. The protocol itself was implemented using the flexible WMP and SnapMAC architectures, allowing fast switching between MAC protocols without downtime.

Initial experiments with a simple cognitive loop have shown a significant performance improvement for both ZigBee and WiFi by eliminating collisions between the two technologies through the activation of the cross-technology TDMA protocol: for a 50% to 50% time division, the 15% of wasted channel time in a legacy configuration can be completely recovered by using the TDMA scheme. Furthermore, we demonstrated the run-time adaptation of the TDMA parameters, showing the capability to dynamically tune the spectrum allocation according to the throughput needs of wireless applications.

The use of OMF/OML experiment control tools as a cognitive network control framework in combination with programmable MAC architectures offers a powerful framework for rapid prototyping and experimental verification of dynamic cognitive control solutions in realistic coexistence scenarios.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work has been partially supported by the EU FP7 Project CREW, Grant Number 258301.

Author details
[1]Ghent University - IBCN - iMinds, Gaston Crommenlaan 8, 9000 Ghent, Belgium. [2]Department of Electical Engineering, Viale delle Scienze ed. 9, 90128 Palermo, Italy.

Received: 28 February 2014 Accepted: 24 November 2014
Published: 6 December 2014

References

1. J Mitola, JGQ Maguire, Cognitive radio: making software radios more personal. Pers. Commun. IEEE. **6**(4), 13–18 (1999)
2. V Bose, V Inc, D Wetherall, J Guttag, in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. Next century challenges: radioactive networks, (1999), pp. 242–248
3. S Pollin, I Tan, B Hodge, C Chun, A Bahai, in *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference On*. Harmful coexistence between 802.15.4 and 802.11: A measurement-based study, (2008), pp. 1–6
4. I Tinnirello, G Bianchi, P Gallo, D Garlisi, F Giuliano, F Gringoli, in *INFOCOM, 2012 Proceedings IEEE*. Wireless MAC processors: programming MAC protocols on commodity hardware (Orlando, FL, USA, 2012), pp. 1269–1277
5. PD Mil, B Jooris, L Tytgat, J Hoebeke, I Moerman, P Demeester, SnapMAC: a generic MAC/PHY architecture enabling flexible {MAC} design. Ad Hoc Netw. **17**, 37–59 (2014)
6. SY Shin, HS Park, WH Kwon, Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b. Comput. Netw. **51**(12), 3338–3353 (2007)
7. L Tytgat, O Yaron, S Pollin, I Moerman, P Demeester, Analysis and experimental verification of frequency-based interference avoidance mechanisms in IEEE 802.15.4. Netw. IEEE/ACM Trans. **PP**(99), 1–1 (2014)
8. L Tytgat, O Yaron, S Pollin, I Moerman, P Demeester, Avoiding collisions between IEEE 802.11 and IEEE 802.15.4 through coexistence aware clear channel assessment. EURASIP J. Wireless Commun. Netw., 137–113715 (2012)
9. R Gummadi, D Wetherall, B Greenstein, S Seshan, in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '07. Understanding and mitigating the impact of RF interference on 802.11 networks (ACM New York, 2007), pp. 385–396
10. J Huang, G Xing, G Zhou, R Zhou, in *Network Protocols (ICNP) 2010 18th IEEE International Conference On*. Beyond co-existence: Exploiting WiFi white space for Zigbee performance assurance (Kyoto, Japan, 2010), pp. 305–314
11. X Zhang, KG Shin, in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. MobiHoc '11. Enabling coexistence of heterogeneous wireless systems: case for Zigbee and WiFi (ACM New York, 2011), pp. 6–1611
12. JM C-Liang, NB Priyantha, J Liu, A Terzis, in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. SenSys '10. Surviving Wi-Fi interference in low power Zigbee networks (ACM New York, 2010), pp. 309–322
13. K Lakshminarayanan, S Sapra, S Seshan, P Steenkiste, in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '09. Rfdump: An architecture for monitoring the wireless ether (ACM New York, 2009), pp. 253–264
14. F Hermans, L-A Larzon, O Rensfelt, P Gunningberg, A lightweight approach to online detection and classification of interference in 802.15.4-based sensor networks. SIGBED Rev. **9**(3), 11–20 (2012)
15. R Zhou, Y Xiong, G Xing, L Sun, J Ma, in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*. MobiCom '10. Zifi: Wireless LAN discovery via ZigBee interference signatures (ACM New York, 2010), pp. 49–60
16. S Rayanchu, A Patro, S Banerjee, in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. Airshark: Detecting non-WiFi RF devices using commodity WiFi hardware (ACM New York, 2011), pp. 137–154
17. R Gummadi, H Balakrishnan, S Seshan, in *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. Metronome: coordinating spectrum sharing in heterogeneous wireless networks (Bangalore, India, 2009), pp. 1–10
18. S Gollakota, F Adib, D Katabi, S Seshan, in *Proceedings of the ACM SIGCOMM 2011 Conference*. SIGCOMM '11. Clearing the RF smog: making 802.11n robust to cross-technology interference (ACM New York, 2011), pp. 170–181
19. The GNURadio Software radio. http://gnuradio.org. Accessed Jan 2014
20. WARP Project - Wireless Open-Access Research Platform. http://warp.rice.edu. Accessed Jan 2014
21. MC Ng, KE Fleming, M Vutukuru, S Gross, M Arvind, H Balakrishnan, in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ANCS '10. Airblue: A system for cross-layer wireless protocol development (ACM New York, 2010), pp. 4–1411
22. J Jun, P Peddabachagari, M Sichitiu, in *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium On*. Theoretical maximum throughput of IEEE 802.11 and its applications (MA, USA, 2003), pp. 249–256
23. D Croce, P Gallo, D Garlisi, F Giuliano, S Magione, I Tinnirello, *Errorsense: Characterizing wifi error patterns for detecting ZigBee interference*. Technical report, Department of Electrical Engineering, Universita di Palermo, Italy
24. EU FP7 CREW project. http://www.crew-project.eu. Accessed Jan 2014
25. G Bianchi, I Tinnirello, in *Proceedings of the 8th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*. WiNTECH '13. One size hardly fits all: towards context-specific wireless mac protocol deployment (ACM New York, 2013), pp. 1–8
26. D Croce, N Galioto, D Garlisi, C Giaconia, F Giuliano, T Ilenia, in *Wintech '14*. Busybee: low rate WiFi-ZigBee communications without gateways (Maui, Hawaii, 2014)
27. w-iLab.t: wireless facilities. http://ilabt.iminds.be/wilabt. Accessed Jan 2014
28. RM090: Ultra low power IEEE 802.15.4 compliant wireless sensor module. http://www.rmoni.com/en/products/hardware/rm090. Accessed Jan 2014

doi:10.1186/1687-1499-2014-212
Cite this article as: De Valck *et al.*: **Exploiting programmable architectures for WiFi/ZigBee inter-technology cooperation.** *EURASIP Journal on Wireless Communications and Networking* 2014 **2014**:212.